Solving an Industrial Oven Scheduling Problem with a Simulated Annealing Approach

Marie-Louise Lackner, Nysret Musliu, and Felix Winter

Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling, DBAI, TU Wien, Favoritenstraße 9, 1040 Vienna, Austria
marie-louise.lackner@tuwien.ac.at, nysret.musliu@tuwien.ac.at, felix.winter@tuwien.ac.at

1 Introduction

In times of a climate crisis, reducing industrial energy consumption has become all the more important. In this paper, we are concerned with the hardening process of electronic components in specialised heat treatment ovens, a highly energy-intensive task. The energy consumption of this process can be reduced by grouping compatible jobs into batches for simultaneous processing.

Recently, we formalized this problem as the Oven Scheduling Problem (OSP), an NP-hard parallel batch scheduling problem [4]. The key task of the OSP is to create a feasible assignment of jobs to batches and to find an optimal schedule of these batches on a set of ovens. The creation of batches must be done in such a way that jobs in the same batch have the same attribute as well as compatible minimal and maximal processing times. Moreover, the scheduling of batches needs to respect the jobs' earliest start times, their respective sets of eligible ovens as well as oven capacities. Furthermore, availability times of ovens as well as attribute-dependent setup times between batches need to be considered. The optimization goal of the OSP is to minimize a linear combination of three objectives: cumulative batch processing time, tardiness and setup costs. For a more formal definition of the OSP, we refer the reader to our previous publication [4].

A wealth of scientific papers has investigated batch scheduling problems throughout the past three decades (see. e.g., the surveys [6, 7, 2]). The problems studied so far in the literature typically minimize objectives related to makespan, tardiness or lateness. The OSP differs from these problems as one of its main objectives is to minimize the cumulative batch processing time across all ovens–which is directly related to the energy costs of running the ovens.

In our recent publications introducing the OSP [4, 5], we provided a benchmark set consisting of 80 instances of different sizes (up to 100 jobs), developed CP and ILP models and performed an extensive experimental evaluation of our proposed exact solution methods. Moreover, we developed theoretical lower bounds on the optimum value.

To increase the practical applicability of our previously developed methods for the OSP, two goals need to be pursued. Most of the large benchmark instances with 50 or 100 jobs could not be solved to optimality within the runtime limit of one hour. Firstly, we thus need to improve the solution quality for large instances. Secondly, in practice one often needs to obtain solutions – of not necessarily optimal, but sufficiently good quality – in shorter time than one hour, ideally in just a couple of minutes. In order to fulfill both of these goals, we propose a metaheuristic local search approach based on simulated annealing which we briefly describe in the following section. In Section 3, we then evaluate the results of our preliminary experiments with this new solution approach for the OSP.

2 The Simulated Annealing Approach

Simulated annealing is a metaheuristic local search technique targeted at finding an approximation of the global optimum for optimization problems with large search spaces. The name of this technique comes from the process of annealing in metallurgy and was introduced by Kirkpatrick, Gelatt and Vecchi [3]. Since, simulated annealing has been successfully employed to solve a variety of real-world optimization problems, including batch scheduling problems (see, e.g., [1]).

In the following, we briefly describe the core concept of simulated annealing and our implementation of this technique for the Oven Scheduling Problem. The simulated annealing algorithm starts off at an initial temperature and with an initial solution that we obtain using the construction heuristic we presented in our previous work [4, 5]. At every iteration step, one of four neighborhood types is chosen uniformly at random, and a candidate solution in this neighborhood is chosen by applying a random move to the current solution. The four neighborhood moves we propose are the following: (i) *Move Job to Batch:* select a job and add it to an existing batch that is compatible, (ii) *Create New Batch from Job:* select a job, remove it from its current batch, create a new batch consisting of this single job and insert this batch somewhere in the current schedule, (iii) *Move Batch:* select an entire batch and insert it at a new position of the current schedule, (iv) *Swap Consecutive Batches*: select two consecutive batches on the same oven and swap them.

The candidate solution obtained in this way is accepted if its solution cost is an improvement over the current solution. In case the candidate solution is a deterioration of the current solution, it is also accepted if the *acceptance criterion* is met. This acceptance criterion depends both on the current temperature and the relative size of the deterioration; we use the metropolis criterion [3] as acceptance function. After this step, the temperature is reduced according to a cooling scheme. This procedure is iterated until the runtime limit is reached and returns the best solution found.

3 Preliminary Experimental Evaluation

In order to assess the viability of the proposed local search approach using simulated annealing, we implemented a preliminary version of the algorithm. Based on manual parameter tuning, we set the initial acceptance rate to 50% and the minimum temperature to 10^{-10}. For a given instance, the initial temperature is chosen so that moves are accepted with probability equal to the initial acceptance rate at the beginning of the local search process. This is done by creating a sample of moves from the initial solution. Moreover, instead of using a fixed cooling rate, the cooling scheme is chosen adaptively for every run of the algorithm: the cooling is done in such a way that the minimum temperature is reached at the same time as the runtime limit. This is ensured by calculating a new, reduced temperature after every iteration step, based on the current average time required per iteration. We performed a series of experiments on our benchmark set [4] consisting of 80 randomly generated instances.[1] The experiments were run on single cores, using a computing cluster with 10 identical nodes, each having 24 cores, an Intel(R) Xeon(R) CPU E5–2650 v4 @ 2.20GHz and 252 GB RAM.

	exact methods	**simulated annealing**	
runtime limit per run	1 hour	30 seconds	5 minutes
# best results	54	56	**77**
# provably optimal results	**37**	36	**37**

Table 1: Overview of the preliminary computational results on the benchmark set consisting of 80 instances.

An overview of our preliminary results can be found in Table 1. In this table, we compare the quality of solutions obtained with the exact methods presented in our previous work [4] with those obtained with the simulated annealing approach. Our previous experiments (presented in [4]) were run in the same computing environment as the ones in this paper. For the exact methods, we use the overall best (minimal) solution cost per instance obtained in [4]. To be precise, this is the best result obtained by any of the 53 exact methods[2] within a runtime limit of one hour. For the simulated annealing approach, we chose to run our algorithm 5 times per instance and use the best result for the comparison. This repetition can be advantageous due to the non-deterministic nature of simulated annealing. We considered two runtime limits: 30 seconds per run (2.5 minutes in total) and 5 minutes per run (25 minutes in total).

The row labeled "# best results" in the table displays the number of instances for which overall best solutions could be achieved (out of 80). The row labeled "# provably optimal results" shows the number of obtained optimal solutions, i.e., solutions for which one of the exact solutions methods could prove optimality. As one can see, running the simulated annealing approach with merely 30 seconds

[1] The benchmark set is publicly available at https://cdlab-artis.dbai.tuwien.ac.at/papers/ovenscheduling/OSPrandominstances/.

[2] This number results from the combination of different models, solvers and search strategies as well as a warm-start option.

per run already allows us to obtain a similar solution quality as with the best exact methods and a runtime limit of one hour. Increasing the runtime of the simulated annealing approach to 5 minutes per run, leads to best results for nearly the entirety of the benchmark set (77 of 80 instances). Furthermore, regarding the instances for which provably optimal solutions could be found by the exact methods, the simulated annealing approach with 5 minutes runtime also finds optimal solutions for every one of these 37 instances; with 30 seconds runtime, optimal solutions can be found for all except one of these instances.

We note that the compared methods were all capable of finding solutions for the entire benchmark set (80 instances). Moreover, both the exact methods and the simulated annealing approach were always capable of improving the (initial) solution provided by the construction heuristic [4].

In Table 2, we take a closer look at the results obtained for the 43 instances of the benchmark set for which none of the exact methods could prove optimality within the runtime limit of one hour. For every one of these instances, we compute the relative improvement ri in % of the simulated annealing approach over the best exact approach. For the simulated annealing approach, we take the best result of 5 runs each having a runtime limit of 5 minutes. In this table, we group the instances by their number of jobs (20, 50 or 100 jobs) and by the value of the relative improvement. Overall, the improvement ri is between -0.05% and 1% for 32 (of 43) instances: for 15 instances, the simulated annealing approach finds solutions of the same quality as the exact approach ($ri = 0$), for another 15 instances, simulated annealing delivers slightly better results ($ri \in (0, 1]$), and for 2 instances slightly worse results ($ri \in [-1, 0)$). For the remaining 11 instances, an improvement of the solution cost by more than 1% was possible, for 7 of which the improvement was larger than 10%. Note that for none of the 43 instances, the solution quality of the simulated annealing approach was significantly worse than the best exact result: there are no instances with $ri < -1\%$.

It is important to note that the size of the instance has a major impact on the improvement made by the simulated annealing approach: for all of the smaller instances with 20 jobs, the solution quality could not be improved; for the instances with 50 jobs, an improvement was possible for 7 out of 17 instances; for the instances with 100 jobs, the solution quality was improved for 19 out of 20 instances. A possible explanation is that the solutions found by the exact methods for the smaller solutions might by optimal–even though no optimality proof could be delivered within the runtime limit of one hour–or very close to the global optimum.

4 Future Work

The preliminary results obtained with the proposed simulated annealing approach on our benchmark set already look promising. As a next step, we plan to configure our algorithm by using automated parameter configuration tools for parameters such as the probabilities of neighborhoods, initial acceptance rate

# jobs	# instances	# instances with relative improvement *ri*					avg *ri*	max *ri*
		$[-1,0)$	$=0$	$(0,1]$	$(1,10]$	>10		
20	6	0	6	0	0	0	0%	0%
50	17	1	9	**6**	0	**1**	0.65%	10.45%
100	20	1	0	**9**	**4**	**6**	7.15%	28.01%
total	43	2	15	**15**	**4**	**7**	3.58%	28.01%

Table 2: Overview of the relative improvement *ri* (in %) achieved by the simulated annealing approach over the best exact approach. Numbers in bold font indicate instances for which an improvement over the best exact result could be achieved. Results are only displayed for the 43 benchmark instances for which no exact method could deliver an optimality proof within the runtime limit of one hour.

and final temperature. Then we plan to conduct an additional evaluation of our simulated annealing approach, including experiments with a longer runtime.

In practice, instances can be even larger than those included in our benchmark set; instances consisting of up to 1500 jobs can be expected. We will therefore also conduct experiments on larger instances than those in our benchmark set. Moreover, we will include the lower bounds obtained by the exact methods [4] as well as the theoretical lower bounds [5] in this evaluation in order to provide a more precise assessment of the solution quality. The theoretical lower bounds, which can be calculated in a few seconds, could also be integrated in a stopping criterion for the simulated annealing approach to reduce the runtime.

Another promising line of research would be to investigate adaptive neighborhood selection. In the current formulation of our simulated annealing approach, the probability of each of the four neighborhoods is constant throughout the solution process. It could however be advantageous to adapt these probabilities based on the current state of the search process. Moreover, it would be interesting to investigate domain-independent hyper-heuristic approaches.

Acknowledgments The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

Bibliography

[1] Damodaran, P., Vélez-Gallego, M.C.: A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times. Expert systems with Applications **39**(1), 1451–1458 (2012)

[2] Fowler, J.W., Mönch, L.: A survey of scheduling with parallel batch (p-batch) processing. European Journal of Operational Research **298**(1), 1–24 (Apr 2022)

[3] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science **220**(4598), 671–680 (May 1983). https://doi.org/10.1126/science.220.4598.671

[4] Lackner, M.L., Mrkvicka, C., Musliu, N., Walkiewicz, D., Winter, F.: Minimizing Cumulative Batch Processing Time for an Industrial Oven Scheduling Problem. In: 27th International Conference on Principles and Practice of Constraint Programming (CP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 210, pp. 37:1–37:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021)

[5] Lackner, M.L., Mrkvicka, C., Musliu, N., Walkiewicz, D., Winter, F.: Exact methods and lower bounds for the oven scheduling problem. Under review (2022), https://arxiv.org/abs/2203.12517

[6] Mathirajan, M., Sivakumar, A.I.: A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. The International Journal of Advanced Manufacturing Technology **29**(9-10), 990–1001 (2006)

[7] Potts, C.N., Kovalyov, M.Y.: Scheduling with batching: A review. European Journal of Operational Research **120**(2), 228–249 (Jan 2000)